# Color

We look into two principal uses of color scales:

- to distinguish unordered qualitative attributes; and
- to represent quantitative data values.

A *color scale*, also called a *colormap*, is a mapping from the range of an attribute's data values to a color range. In the case of continuous color scales, indexes into the color scale are interpolated.

Plotly references on color: continuous color scales, discrete color scales, and built-in color scales.

```
from google.colab import drive
drive.mount('/content/drive')

# datadir = 'data/'
# imagesdir = 'images/'


from IPython.display import Image, display

datadir = '/content/drive/My Drive/Courses/672/Notebooks/Notebooks2023/data/'
imagesdir = '/content/drive/My Drive/Courses/672/Notebooks/Notebooks2023/images/'

def display_images(images, dir=imagesdir):
    for image in images:
        display(Image(dir + image))
```

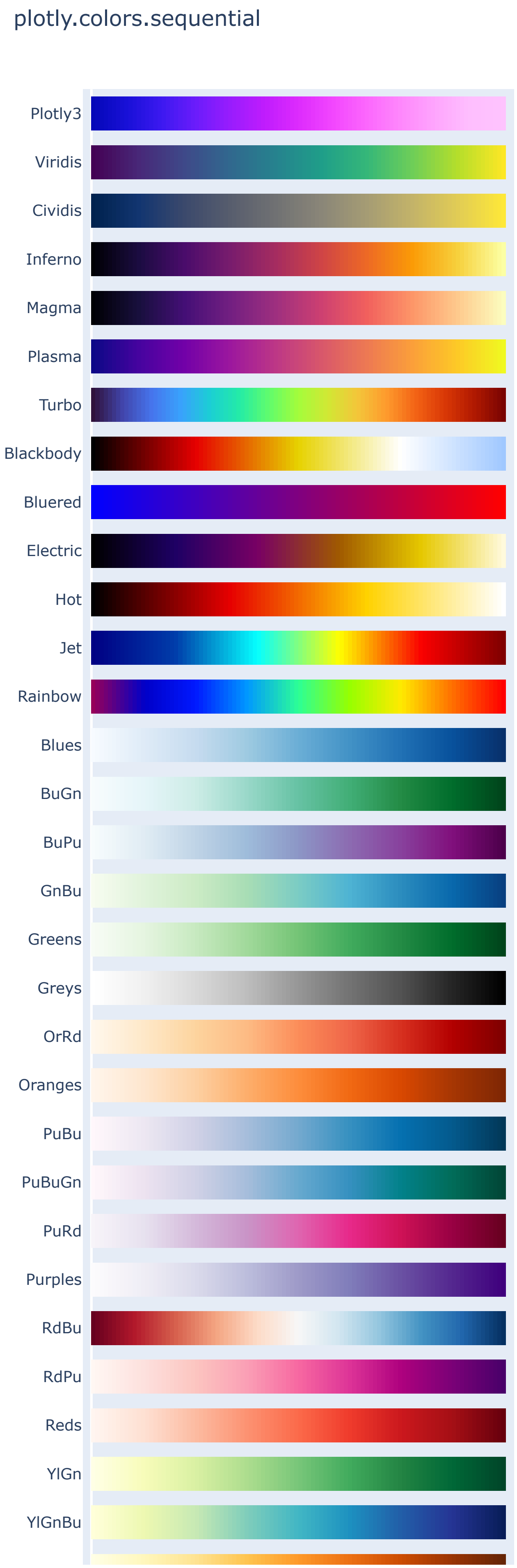
```
Mounted at /content/drive
```

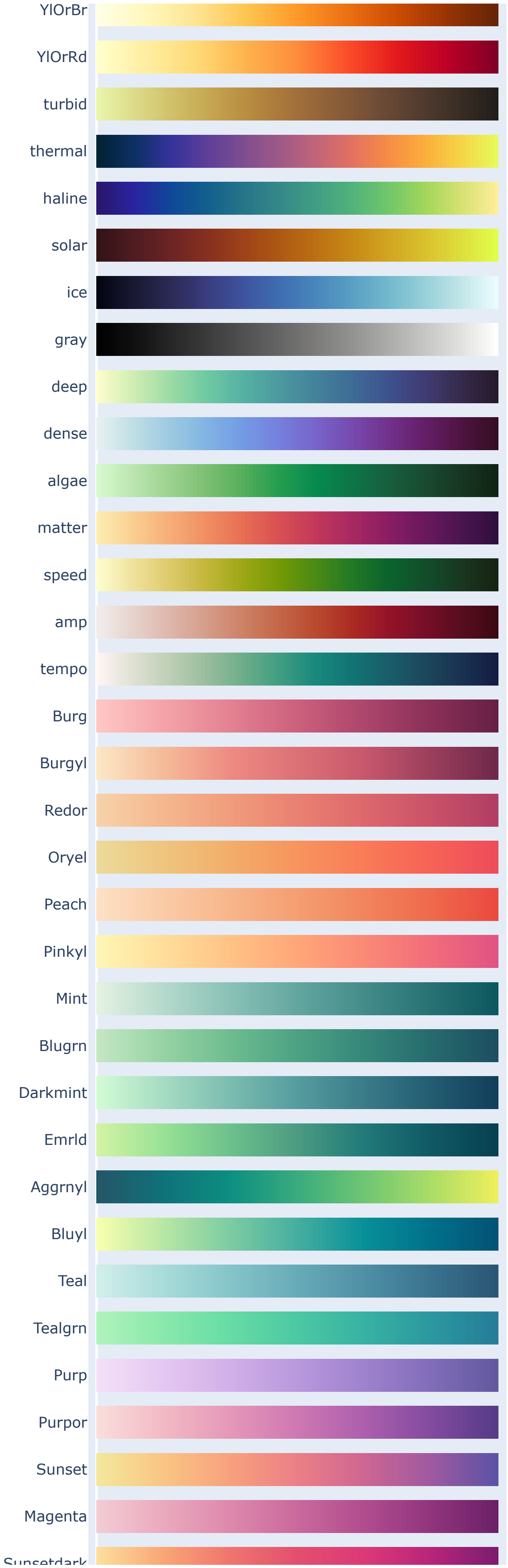
```
import plotly.express as px
import plotly.graph_objects as go
import numpy as np

rendering = None

def show(fig):
  fig.show(rendering=rendering)
```

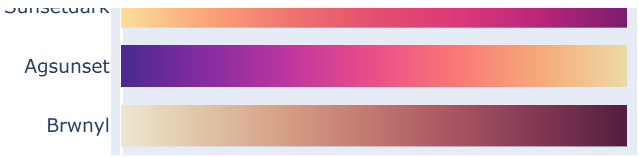
## Continuous color scales

Continuous color scales are used to represent continuous numeric data values. Here we view the built-in continuous color scales.

```
fig = px.colors.sequential.swatches_continuous()
show(fig)
```


plotly.colors.sequential
Plotly3
Viridis
Cividis
Inferno
Magma
Plasma
Turbo
Blackbody
Bluered
Electric
Hot
Jet
Rainbow
Blues
BuGn
BuPu
GnBu
Greens
Greys
OrRd
Oranges
PuBu
PuBuGn
PuRd
Purples
RdBu
RdPu
Reds
YlGn
YlGnBu

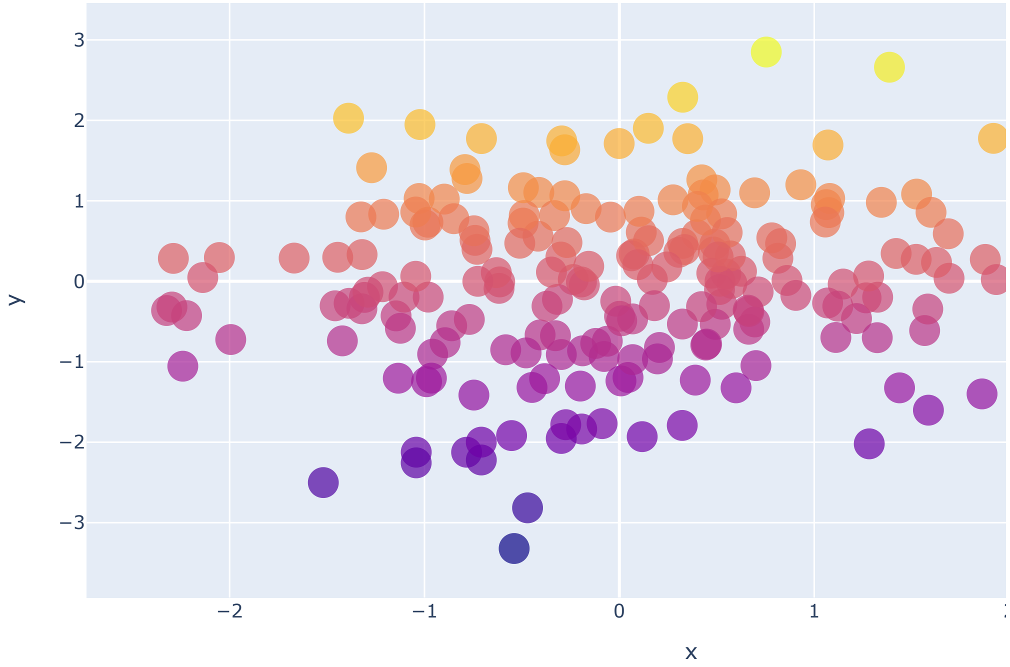

YlOrBr
YlOrRd
turbid
thermal
haline
solar
ice
gray
deep
dense
algae
matter
speed
amp
tempo
Burg
Burgyl
Redor
Oryel
Peach
Pinkyl
Mint
Blugrn
Darkmint
Emrld
Aggrnyl
Bluyl
Teal
Tealgrn
Purp
Purpor
Sunset
Magenta
Sunsetdark



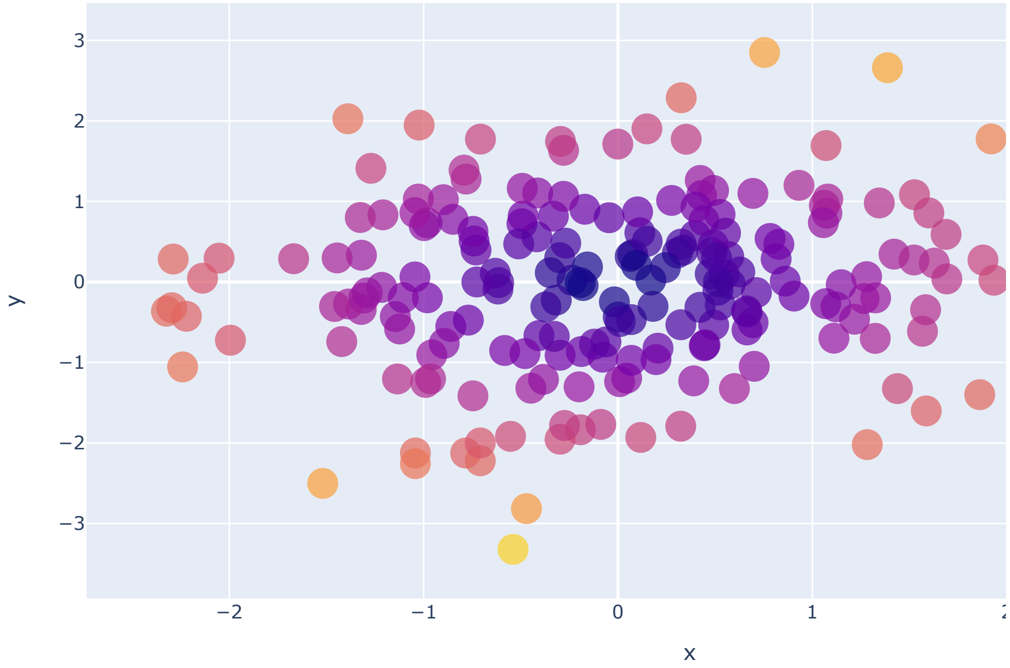
We generate some normally-distributed points in the plane. When we generate a plot, the `color` argument supplies a value for each point corresponding to its color under the current color scale which defaults to *Plotly*.

```
nbr_points = 200
x, y = np.random.normal(0, 1, nbr_points), np.random.normal(0, 1, nbr_points)
```

```
size = 20
fig = px.scatter(x=x, y=y, opacity=0.7, color=y)
fig.update_traces(marker_size=size)
show(fig)
```

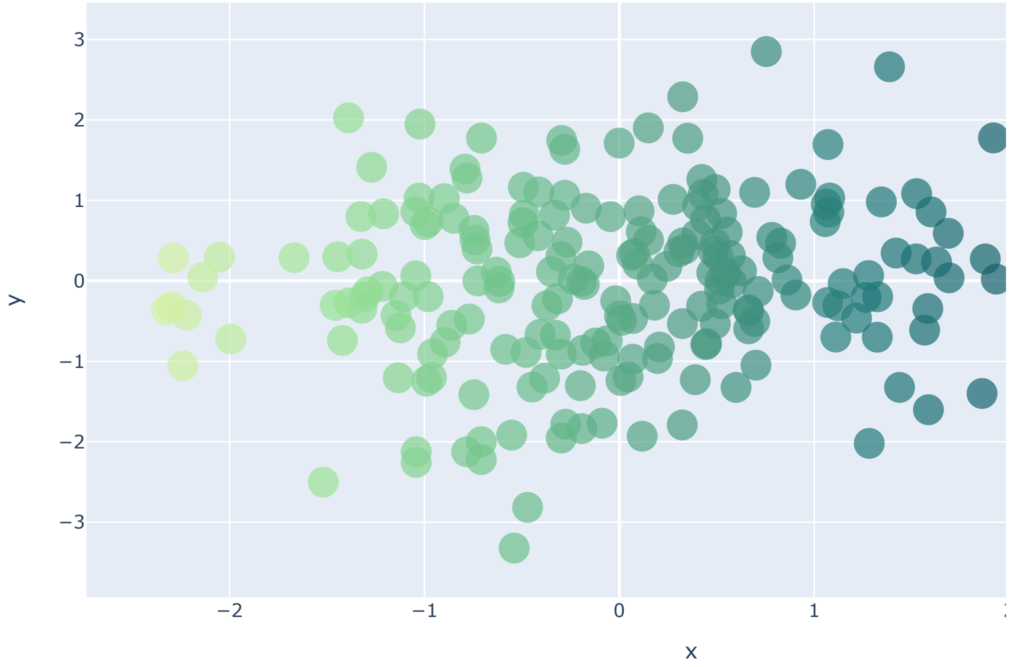


```
def norm(x, y):
  return np.sqrt(x**2 + y**2)
fig = px.scatter(x=x, y=y, opacity=0.7, color=norm(x, y))
fig.update_traces(marker_size=size)
show(fig)
```

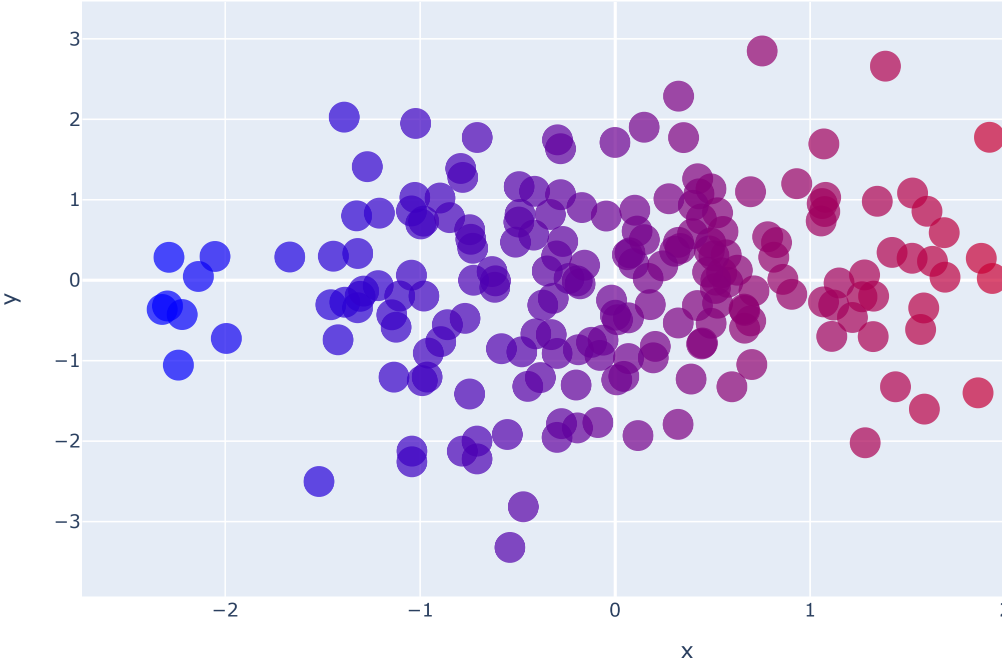


For many plotting functions, we use the `color_continuous_scale` argument to select a different color scale.

```
fig = px.scatter(x=x, y=y, opacity=0.7, color=x, color_continuous_scale=px.colors.sequential.Emrld)
# fig = px.scatter(x=x, y=y, opacity=0.7, color=x, color_continuous_scale='Rainbow')
fig.update_traces(marker_size=size)
show(fig)

```

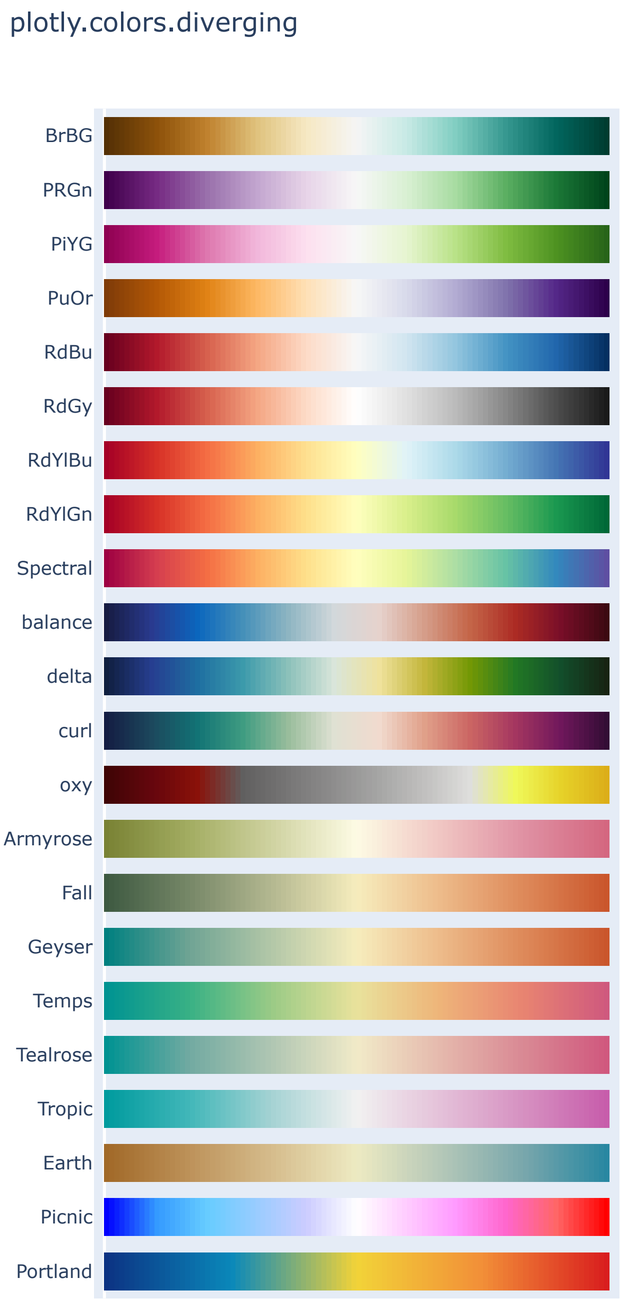


```
fig = px.scatter(x=x, y=y, opacity=0.7, color=x, color_continuous_scale='Bluered')
fig.update_traces(marker_size=size)
show(fig)
```

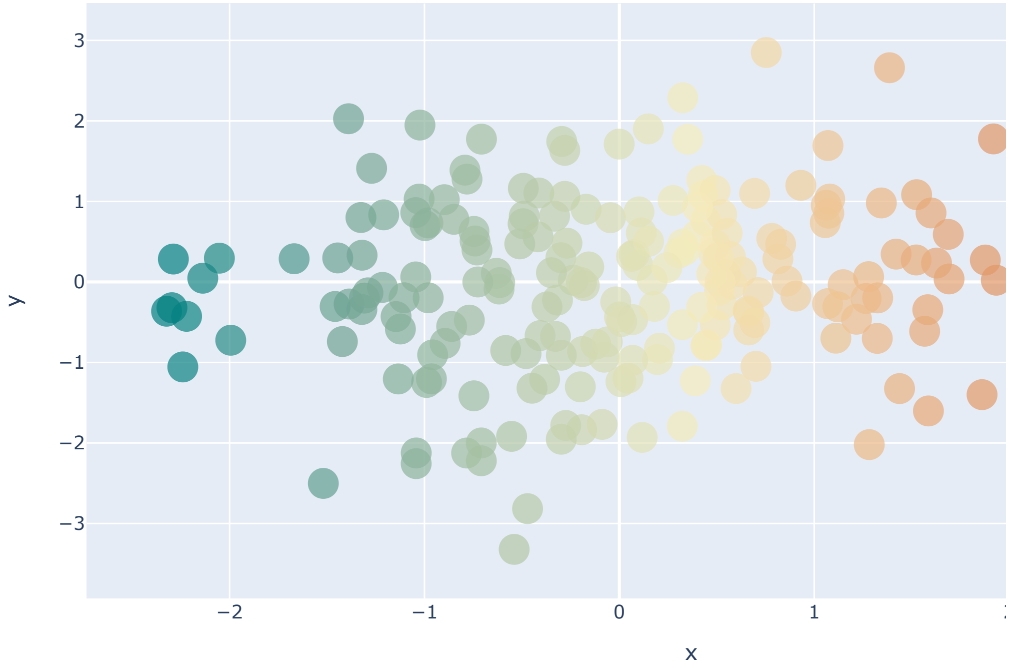


Continuous color scales are used when the data values contain no special *zero* value. We use a *diverging* color scale when we want to distinguish between values less than zero and values greater than zero.

```
show(px.colors.diverging.swatches_continuous())
```

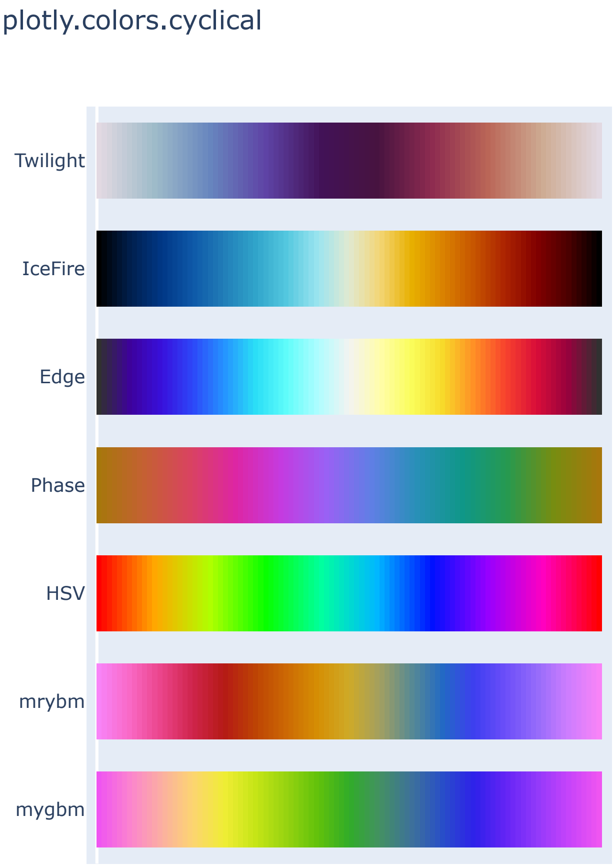


```
fig = px.scatter(x=x, y=y, opacity=0.7, color=x, color_continuous_scale='Geyser')
fig.update_traces(marker_size=size)
show(fig)
```

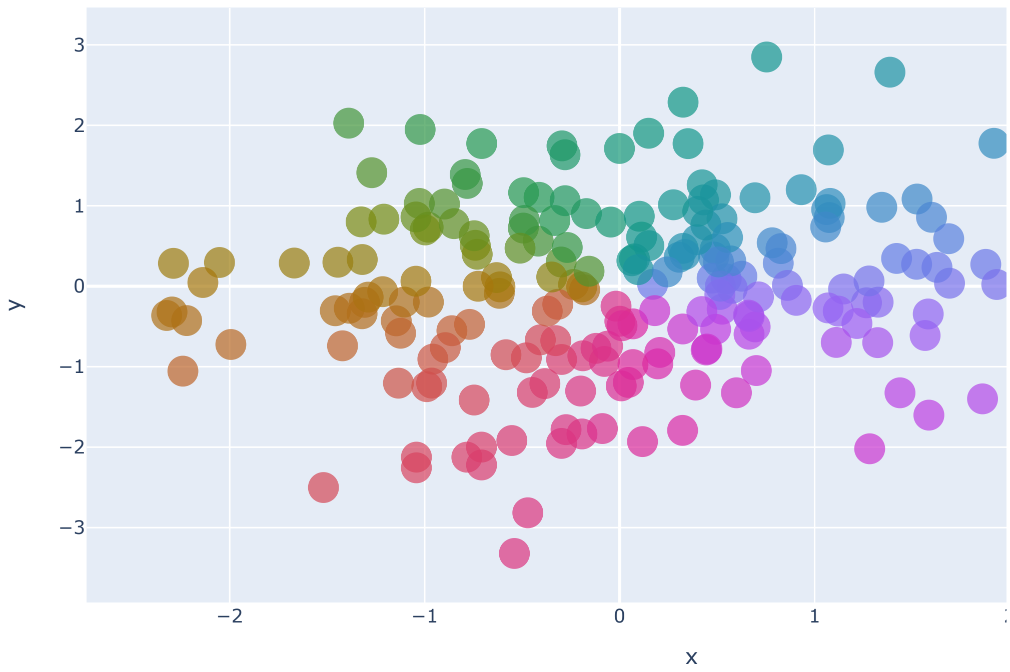


*Cyclic* color scales are used for data that is cyclic.

```
show(px.colors.cyclical.swatches_continuous())
```



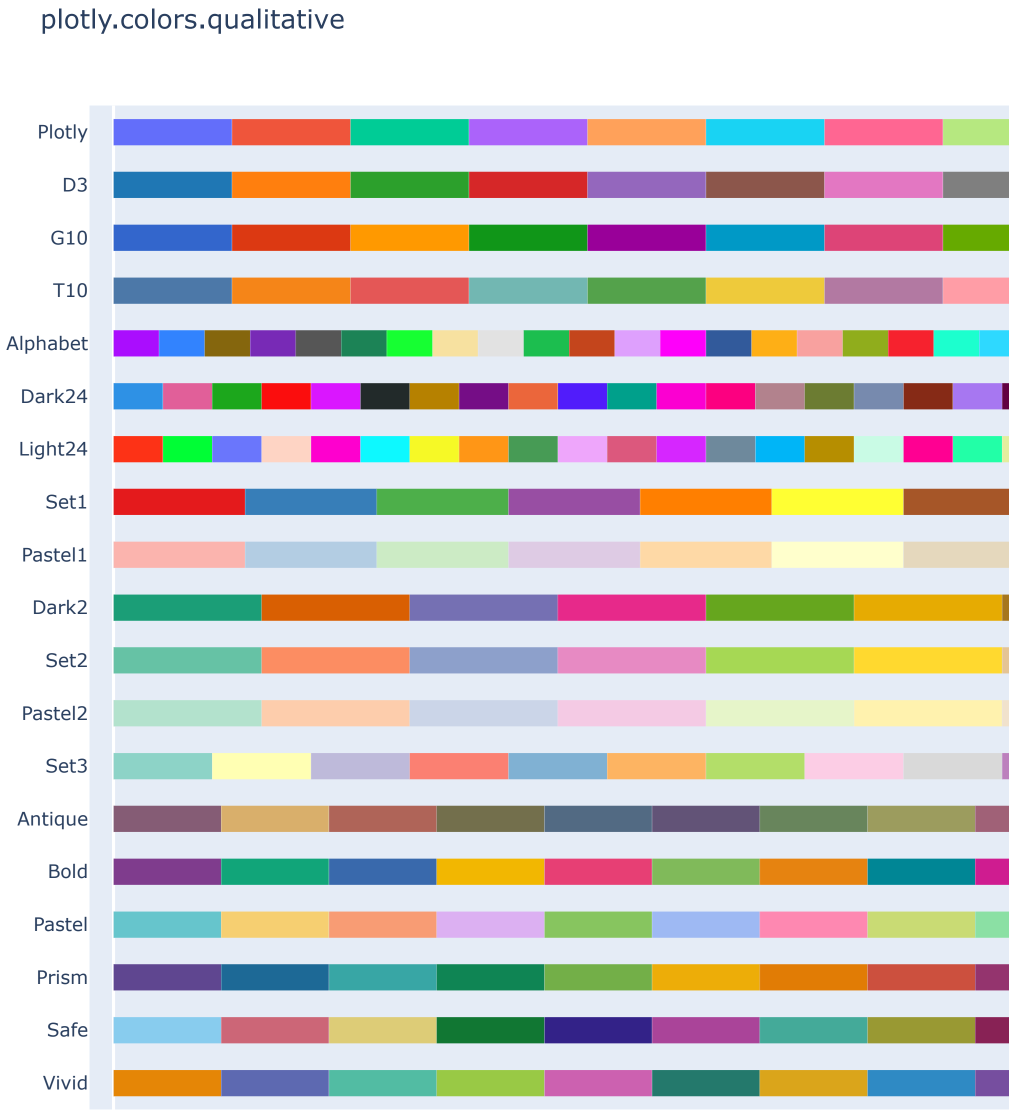
```
def angle(x, y):
  return np.arctan2(x, y)

fig = px.scatter(x=x, y=y, opacity=0.7, color=angle(y,x), color_continuous_scale=px.colors.cyclical.Phase)
fig.update_traces(marker_size=size)
show(fig)
```



## Discrete color scales

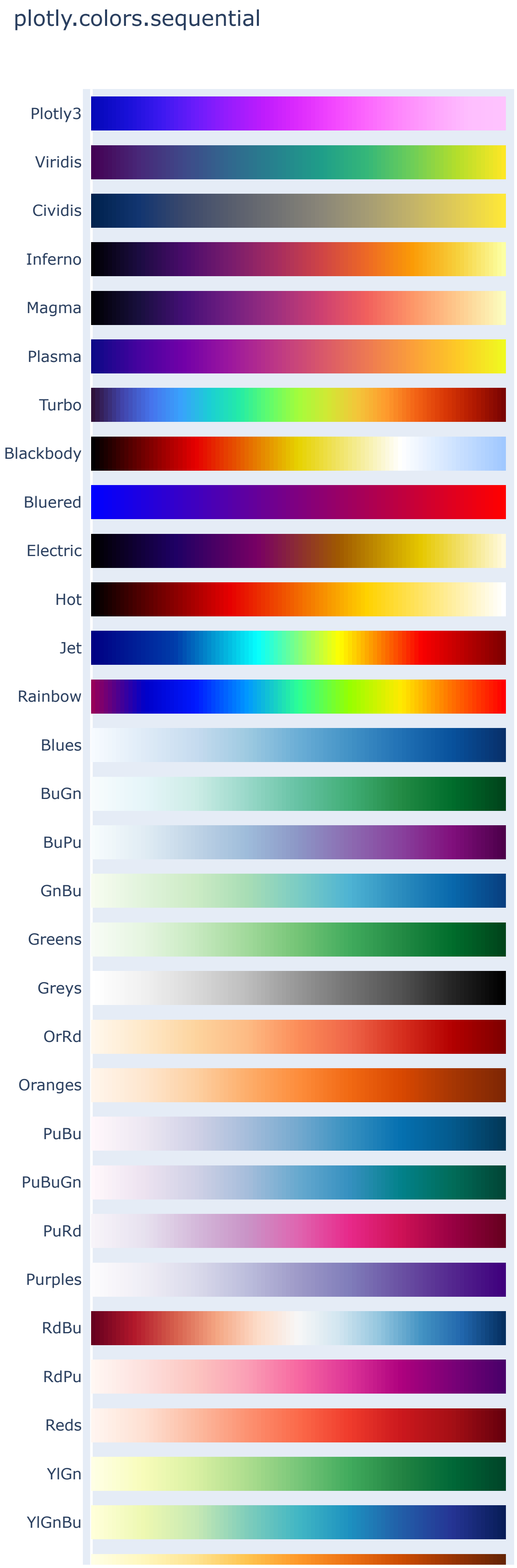
Discrete color scales are used to represent categorical or discrete data values.

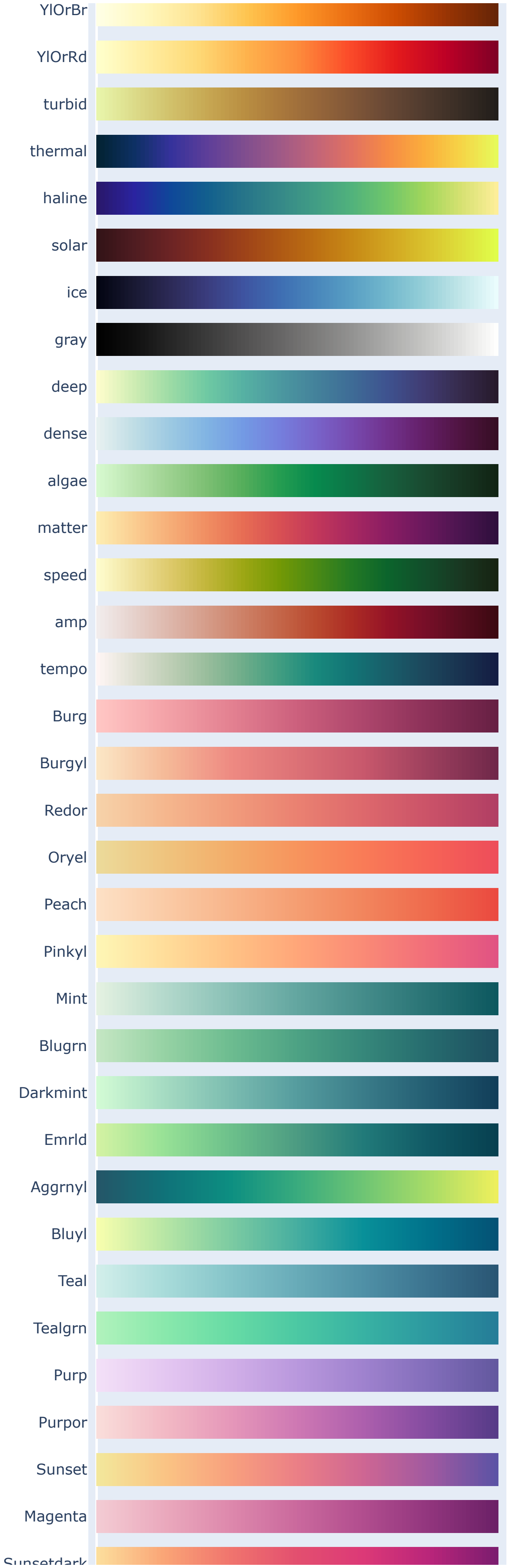
```
show(px.colors.qualitative.swatches())
```

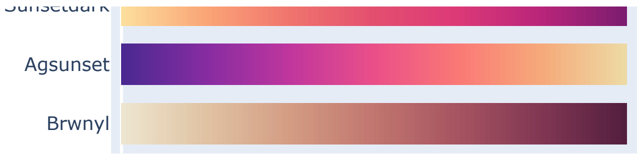


We also have discrete versions of the continuous colormaps we've already seen.

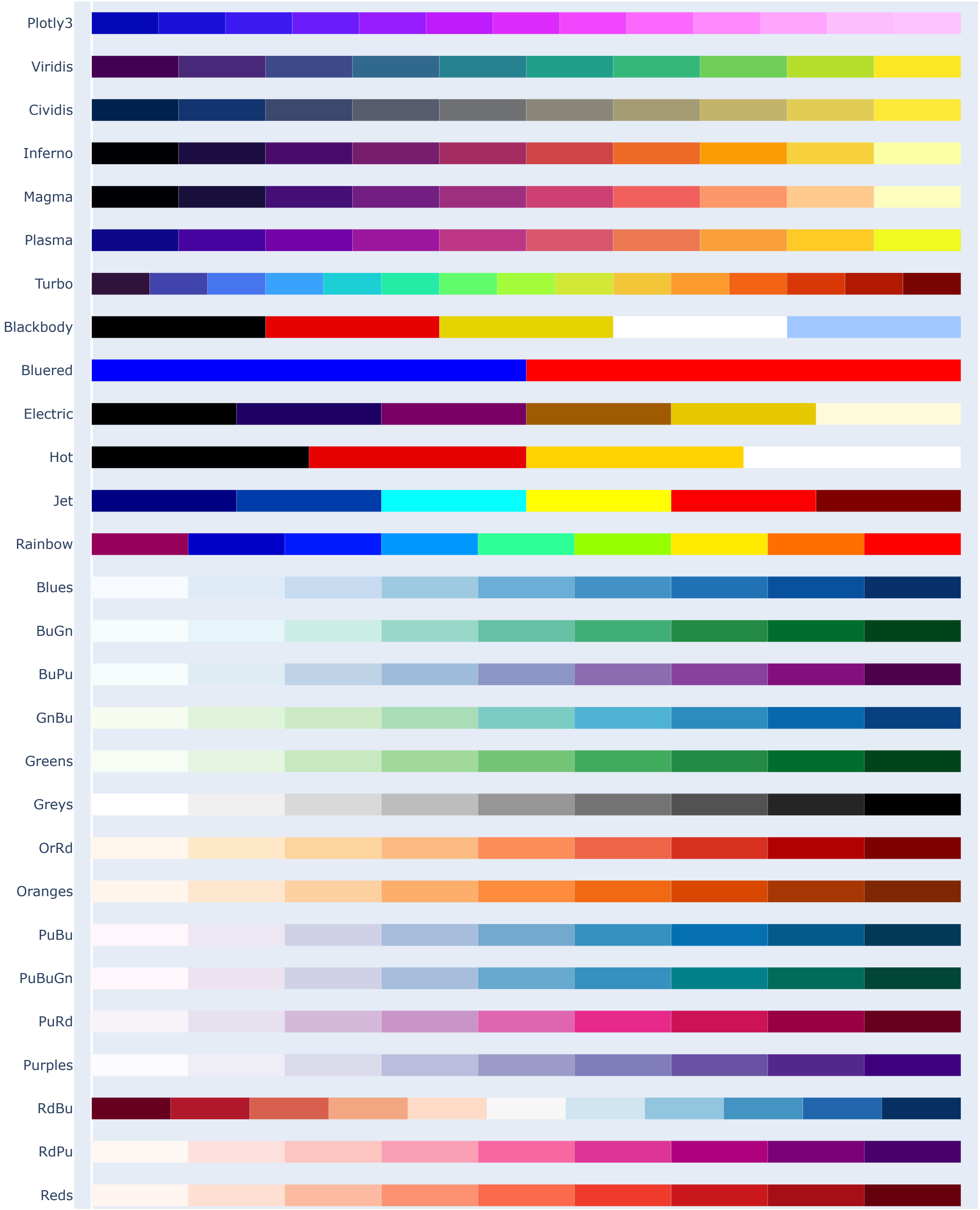
```
show(px.colors.sequential.swatches_continuous())
show(px.colors.sequential.swatches())
```

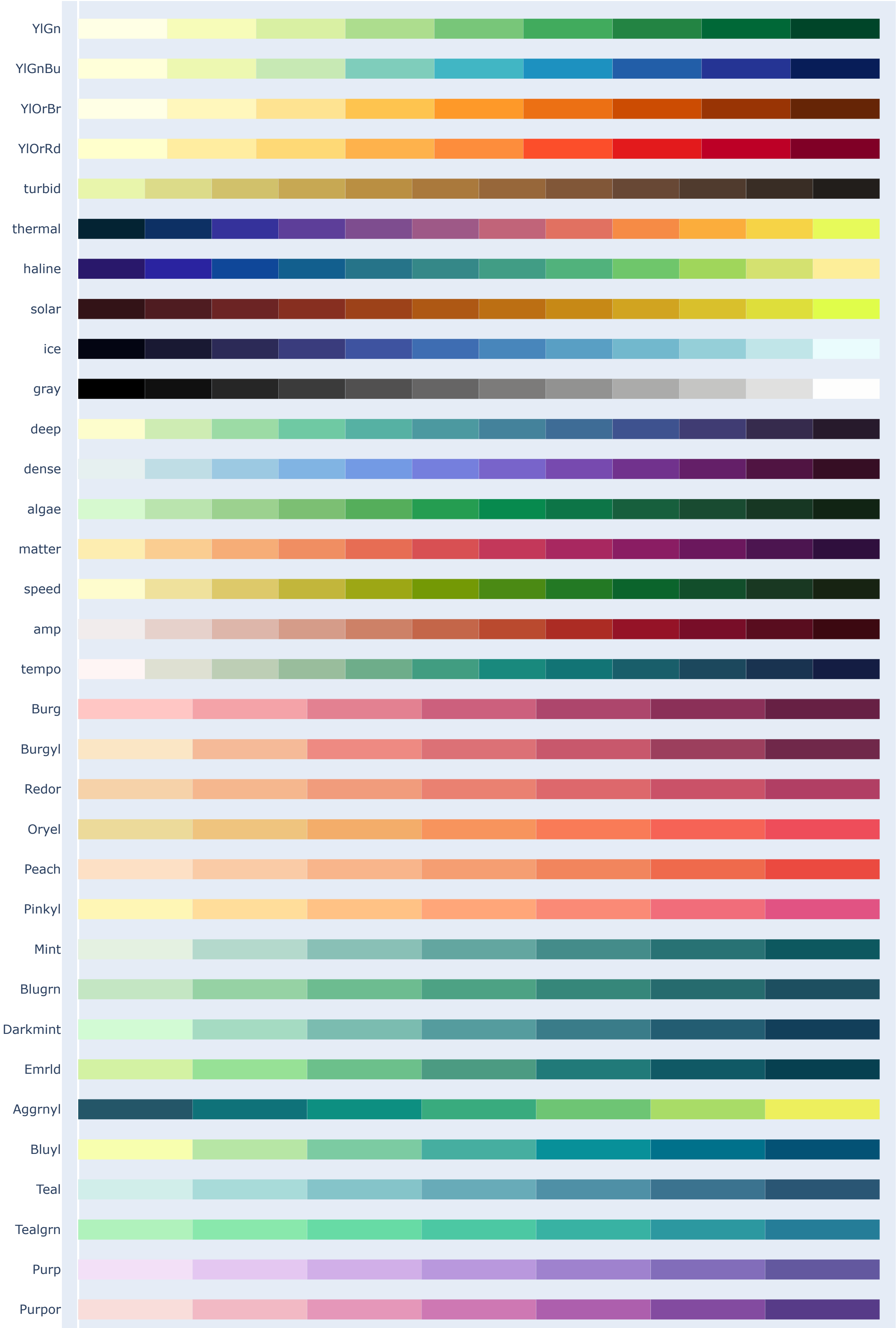

plotly.colors.sequential
Plotly3
Viridis
Cividis
Inferno
Magma
Plasma
Turbo
Blackbody
Bluered
Electric
Hot
Jet
Rainbow
Blues
BuGn
BuPu
GnBu
Greens
Greys
OrRd
Oranges
PuBu
PuBuGn
PuRd
Purples
RdBu
RdPu
Reds
YlGn
YlGnBu

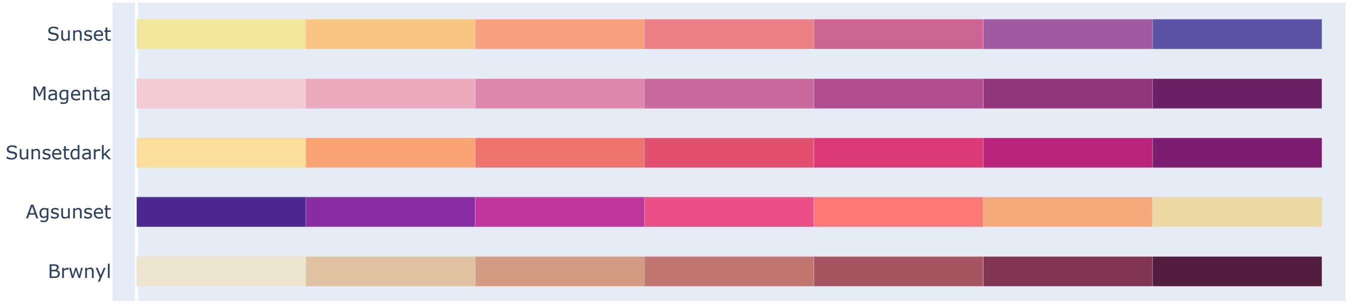

YlOrBr
YlOrRd
turbid
thermal
haline
solar
ice
gray
deep
dense
algae
matter
speed
amp
tempo
Burg
Burgyl
Redor
Oryel
Peach
Pinkyl
Mint
Blugrn
Darkmint
Emrld
Aggrnyl
Bluyl
Teal
Tealgrn
Purp
Purpor
Sunset
Magenta
Sunsetdark



plotly.colors.sequential




YlGn
YlGnBu
YlOrBr
YlOrRd
turbid
thermal
haline
solar
ice
gray
deep
dense
algae
matter
speed
amp
tempo
Burg
Burgyl
Redor
Oryel
Peach
Pinkyl
Mint
Blugrn
Darkmint
Emrld
Aggrnyl
Bluyl
Teal
Tealgrn
Purp
Purpor



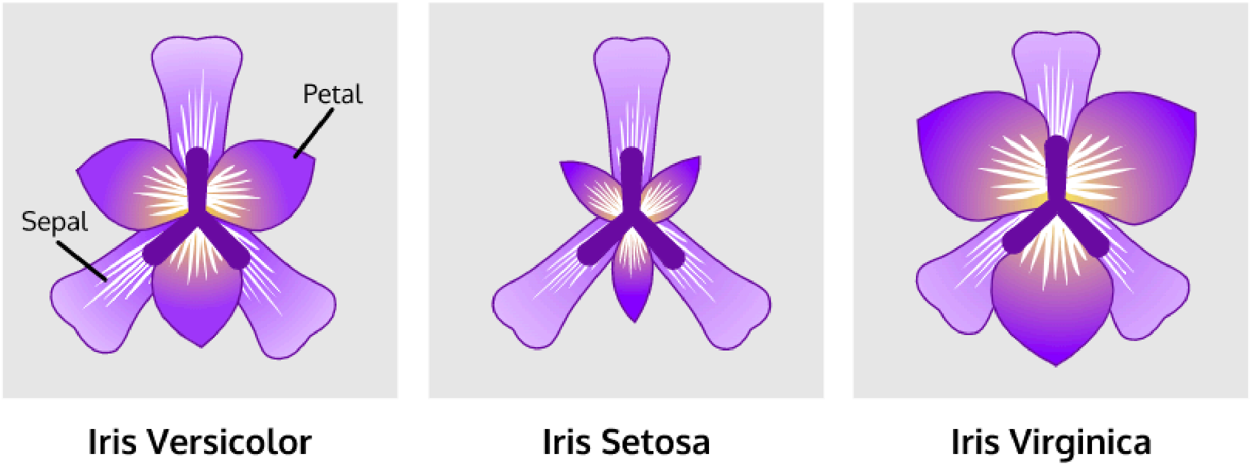
```
df = px.data.iris() # iris is a pandas DataFrame
df
```

| | sepal_length | sepal_width | petal_length | petal_width | species | species_id |
|---|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | setosa | 1 |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | setosa | 1 |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | setosa | 1 |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | setosa | 1 |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | setosa | 1 |
| **...** | ... | ... | ... | ... | ... | ... |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | virginica | 3 |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | virginica | 3 |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | virginica | 3 |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | virginica | 3 |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | virginica | 3 |

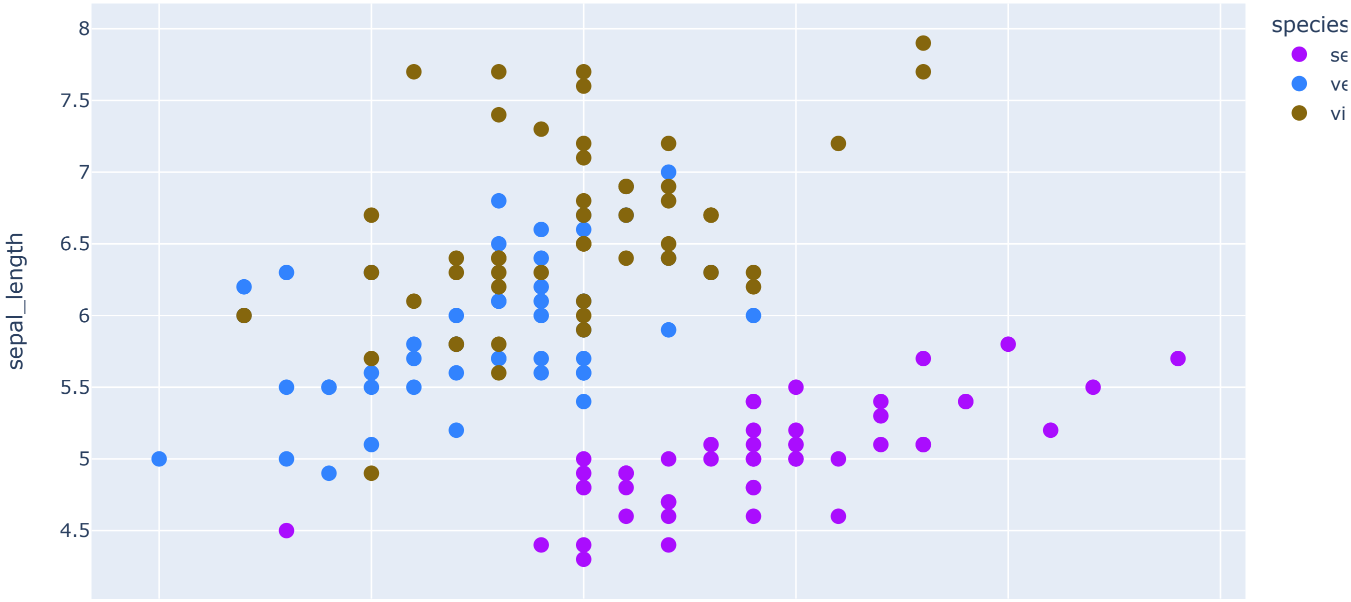
150 rows × 6 columns

Next steps: Generate code with df View recommended plots

```
display_images(['iris.png'])
```



```
# use color to distinguish species (discrete)
fig = px.scatter(df, x="sepal_width", y="sepal_length", color='species', color_discrete_sequence=px.colors.
# some discrete colormaps: color_discrete_sequence=px.colors.qualitative.*: Bold, Alphabet
# fig = px.scatter(df, x="sepal_width", y="sepal_length", color='species', color_discrete_sequence=px.color
# fig = px.scatter(df, x="sepal_width", y="sepal_length", color='species')
# some discrete colormaps: color_discrete_sequence=px.colors.sequential.*: RdBu, Tealgrn, BuPu, PuBu
fig.update_traces(marker_size=10)
show(fig)
```


sepal_length
species
● se
● ve
● vi